Neural Style Transfer

Hanoi University of Science and Technology
Supervisor: Prof. Le Thi Lan and Prof. Vu Thi Thanh Hai
Dương Thị Huyền 20224317
huyen.dt224317@sis.hust.edu.vn
Nguyễn Phạm Trà My 20224327
my.npt224327@sis.hust.edu.vn

c1. Introduction

Neural Style Transfer (NST) is an innovative technique that blends the content of one image with the artistic style of another to produce visually striking stylized images. First introduced by Gatys et al., this method leverages a pre-trained convolutional neural network (CNN) to extract and refine deep features that represent content and style. By bridging the realms of art and technology, NST demonstrates the creative potential of neural networks in image synthesis.

In this project, we reimplement Gatys's algorithm to validate its effectiveness in generating high-quality stylizations, reaffirming its position as a foundational approach in NST. Additionally, we explore the impact of adjusting key parameters, such as loss weights, on the output quality, uncovering opportunities for further refinement and enhancement of NST techniques.

The complete implementation code is available in the accompanying GitHub repository.

2. Related Work

A comprehensive overview in the field of NST can be found in Jing et al. (2018).

The original NST algorithm was first introduced by Gatys et al.(2015). Their work demonstrated that deep features extracted from pre-trained CNNs, specifically the VGG network, could effectively capture content and style information from images. By defining separate loss functions for content and style, their method optimizes a generated image to achieve a visually compelling blend of an arbitrary style image onto an arbitrary content image. However, the original NST has several drawbacks despite its groundbreaking nature. The method requires iterative optimization for each stylized image, making it computationally expensive and time-consuming. It relies heavily on high-performance GPUs, limiting its accessibility for real-time applications or users with limited computational resources.

Following the foundational work of Gatys et al. several variations and improvements have been proposed. Johnson et al. (2016) introduced a real-time style transfer by training feedforward networks, reducing the computational cost compared to the iterative optimization approach of Gatys. The model is trained using a dataset of images and corresponding styles, enabling it to transfer style in a single pass through the network. While this method is much faster than Gatys' original approach, it sacrifices some flexibility. It can only transfer the styles it was trained on, meaning it is not capable of transferring arbitrary styles without retraining the model. Compared to Gatys' optimization-based approach, the results may have lower quality, especially when fine details are crucial, because the feed-forward network does not fully explore the content and style space.

Other methods have used generative adversarial networks (GANs) for optimizing NST, like CycleGAN (Zhu et al., 2017), which allows style transfer without needing paired datasets. However, GANs are challenging to train and can suffer from issues like mode collapse, leading to inconsistent results or artifacts. While CycleGAN works with unpaired data, its output quality may not be as high as methods that use paired datasets, often lacking in content detail or producing less polished stylizations.

This project builds directly on the original method by Gatys et al., reimplementing their approach to analyze its core principles and effectiveness. Our work also considers parameter adjustments, such as loss weights, to better understand their influence on the stylization quality and to identify potentials for future exploration.

3. Dataset

In artwork generation and algorithm evaluation, we use a variety of content of style images. The content images represent the scene or object whose structure we aim to preserve, while the style images define the artistic

features, textures, and color schemes to be applied to the content.

As content images, we selected a diverse set of content images, including portraits, landscapes, and architectural photographs, to explore the algorithm's ability to maintain the content structure across different types of images.

As style images, A variety of artistic-style images were used to provide different visual aesthetics. These include famous paintings and abstract artworks. The purpose of using these styles was to evaluate how well the algorithm transfers artistic textures and patterns onto the content images.

Before feeding the images into the model, several preprocessing steps were applied to ensure compatibility with the neural network. All images are resized to a consistent resolution of 512 (if the machine is compatible with GPU) or 128 (if no GPU detected). The pixel values of the images are normalized based on the mean and standard deviation of the ImageNet dataset, which is commonly used for pre-trained models like VGG-19. In addition, the format of images is BGR, as required by the VGG-19 network. As we use Pytorch for code implementation, the images are reshaped to include a batch dimension to match the input format expected by the neural network.

4. VGG-19 Network

The VGG-19 (Simonyan & Zisserman, 2014) network is a deep convolutional neural network (CNN) widely used for image feature extraction and classification tasks. Originally developed by the Visual Geometry Group at Oxford University, VGG-19 is a variant of the VGG architecture that contains 19 layers (16 convolutional layers and 3 fully connected layers). The network is pre-trained on the ImageNet dataset, which enables it to capture high-level semantic features in images that are crucial for style transfer.

4.1. Architecture

The VGG-19 network consists of 19 layers. There are 16 convolutional layers that perform feature extraction at various spatial resolutions, with ReLU activations applied after each convolution. Each layer acts as a filter that detects specific features in the input image, where earlier layers detect low-level features such as edges and textures, while the deeper layers detect more complex and abstract features, such as shape, patterns, and objects. Following some of the convolutional layers, max pooling layers are introduced to reduce the spatial dimension of the feature maps, allowing the network to focus more on abstract representations while also reducing computational complexity. The final three fully connected layers are used for classification tasks; however, for style transfer, only the convolutional layers are used to extract content and style features, while the fully connected layers are ignored.

4.2. VGG-19 in NST

In the context of NST, VGG-19 plays a critical role in extracting the content and style features of the input images. The content of an image refers to the high-level structure and objects in the scene. These features can be extracted from the higher layers of the network, in this project, layer conv_4 is chosen for this task. The style of an image refers to its textures, patterns, and color distributions. Layers conv_1, conv_2, conv_3, conv_4, and conv_5 are used to get the style representation.

5. Deep Image Representation

The VGG network, which was trained for object recognition and localisation and thoroughly described in the original work, was the foundation for the results that were produced. The feature space of a normalized version of the 19-layer VGG network's 16 convolutional and 5 pooling layers was utilized. By adjusting the weights so that the mean activation of each convolutional filter across images and locations equals one, the network was normalized. This re-scaling is possible for the VGG network without affecting its output because it only uses rectifying linear activation functions and doesn't normalize or pool across feature maps. None of the fully connected layers are used. The model is accessible to the public and can be examined using the caffe-framework. It was discovered that for image synthesis, substituting average pooling for the maximum pooling operation produces slightly more pleasing results; as a result, the images displayed were created using average pooling.

Content loss function: The goal of content loss is to align the transfer image's features with the content image's features. The content loss is calculated as the mean squared difference between content image features and transfer image features for each content feature layer [1].

$$L_{content} = \sum_{l} W_c^l \times \frac{1}{HWC} \sum_{i,j} \left(\widehat{Y}_{i,j}^l - Y_{i,j}^l \right)^2$$

Where:

$\widehat{Y}$ is the predicted feature map for the transfer image.

Y is the predicted feature map for the content image.

W_c^l is the content layer weight for the l^{th}.

H, W, C are height, width, and channels of the feature map.

Identify the names of the layers used for content feature extraction. These features are used to determine content loss. Deeper layers in the VGG-19 network provide more effective training features compared to shallow layers. As a result, specify the fourth convolutional layer for content feature extraction.

Style loss function: The goal of style loss is to match the texture of the transfer image to the texture of the style image. This is achieved by representing the style of an image as a Gram matrix and calculating the mean squared difference between the Gram matrices of the style and transfer images.

$$G_{\hat{Z}} = \sum_{i,j} \widehat{Z_{i,j}} \times \widehat{Z_{j,i}}$$

$$G_{Z} = \sum_{i,j} \widehat{Z_{i,j}} \times \widehat{Z_{j,i}}$$

$$L_{style} = \sum_{l} W_s^l \times \frac{1}{(2HWC)^2} \Sigma \left(G_{\hat{Z}}^l - G_Z^l \right)^2$$

Adjust the weights assigned to the style feature extraction layers. Use lower weights for images with simpler styles and increase the weights for images that have more complex styles.

Total Loss Function: Content loss and style loss are weighted and combined to find the total loss. The weight factors for content loss and style loss are α and β, respectively.

$$L_{total} = \alpha \times L_{content} + \beta \times L_{style}$$

Specify the weight factors alpha and beta for content loss and style loss. The ratio of alpha to beta should be around 1e-3 or 1e-4.

6. Evaluation and Discussion

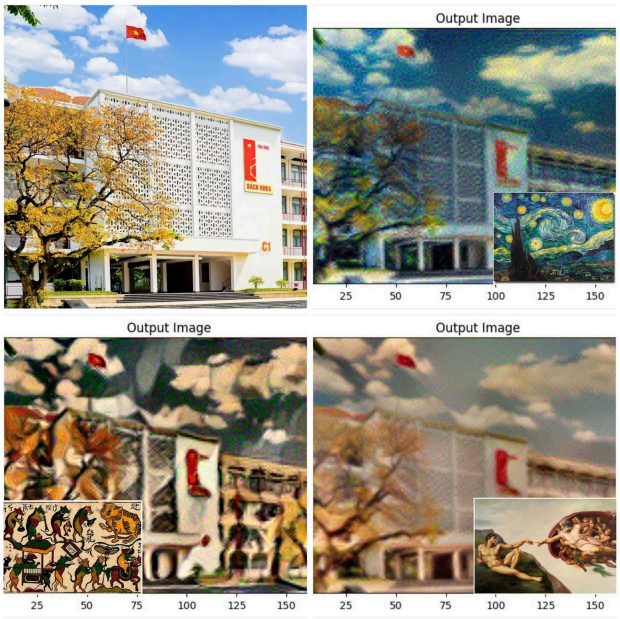
As mentioned above, we use the default weights of VGG-19 and the choice of content and style extraction layers agrees with Gatys et al. The weights $\{w_s^{[l]}\}$, $\{w_c^{[l]}\}$ are all set to 1. Content weight α and style weight β are set to 1 and 1000 correspondingly. The loss function chosen is L-BFGS, with the number of iterations initially set to 1000; however, steps can vary with different style transfers and will be adjusted based on convergence. We have settings for hyperparameters defined as in the table below.

Table 1: Hyperparameter settings

Content layer	'conv4_2'
Style layers	'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1', 'conv5_1'
$\{w_c^{[l]}\}$	1
$\{w_s^{[l]}\}$	1, 1, 1, 1, 1
(α, β)	(1, 10000)

Besides the default setting of hyperparameters, we experiment with the combination of the content and the style image by varying the style weight hyperparameters β . These parameters are tested with values ranging from 10000 to 8000, and then to 5000. The corresponding results are presented in Figure 1.

In addition to that, we test algorithms with arbitrary styles and content images. Figure. 2 shows the application of landscape images to various styles. With the default hyperparameter settings, we can obtain acceptable results.



7. Conclusion

In this project, we successfully reimplemented the neural style transfer (NST) method proposed by Gatys et al. The experiment showcased the algorithm's ability to generate stylized images by blending content and style representations. By adjusting hyperparameters such as the style weight, we observed how these changes impacted the output, providing valuable insights into the trade-offs between content preservation and style transfer. Gatys's method supports arbitrary style transfer, and yields acceptable results. However, the output images are limited in resolution and contain residual noise from random image initialization. NST comes with a significant computational cost, requiring approximately 20 minutes of CPU runtime for a 256x256 image. Moreover, each pair of content and style images necessitates custom hyperparameter tuning to achieve the most visually appealing output.

For future work, we plan to explore further optimization of the hyperparameter settings to refine the balance between the content and style in the generated images. More advanced techniques, such as perceptual loss function or multi-scale features, could improve the preservation of finer content details while enhancing the stylistic effects. Another key direction is optimizing the computational efficiency of the algorithm to reduce the high runtime cost. Fast Neural Style Transfer introduced by Johnson et al. can be a promising direction with its utilization in feedforward networks to enable for real-time style transfer. By exploring these directions, we hope to improve both the efficiency and quality of neural style transfer, making it more suitable for real-time applications and opening up new possibilities for artistic image generation.

8. References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style." Preprint, submitted September 2, 2015. https://arxiv.org/abs/1508.06576